アセンブラに対する教育と作成を支援するツールに関する研究

アセンブラに対する教育と作成を支援するツールに関する研究

RESEARCH ON TOOLS TO SUPPORT ASSEMBLER EDUCATION AND CREATION

岩本　和也
Kazuya　Iwamoto
指導教員　和田　幸一

法政大学大学院理工学研究科応用情報工学専攻修士課程

The Hack system [1] is supposed to be a system that allows even inexperienced students to understand the behavior of computer hardware and software. However, when we make the Hack assembler using this system, We feel it seems to be difficult for beginners to understand how to make it. We need several tools where the conversion process from input to output can be visualized. In this thesis, we thought that its improvement would lead to educational support, and we renovated this system. As a concrete improvement plan, we propose a new system that visualizes the structure and operations of the Hack assembler. We also compared the Hack assembly language with general assembly language. These tools will make it easy for even beginner assemblers to understand.

***Key Words** : Hack system, Computed aided education, Assembler*

１．序論

情報系の学科において，ハードウェア，アーキテクチャ，オペレーティングシステム，プログラミング，コンパイラ，データ構造，アルゴリズムやエンジニアリングなどはカリキュラムとして組み込まれている．これらは計算機科学におけるコンピュータシステムに現れる諸概念であり，コンピュータ技術が発達して複雑化するにつれて，各々を深く理解することは難しくなっている．コンピュータシステムを理解するには，これら密接する諸概念の理解が求められる．しかしながら，諸概念は互いに関連しており，コンピュータシステム全体を理解することは難しい.

そこで，計算機科学の教育を目的として The Elements of Computing Systems[1]が開発された．その教育をはじめとした多方面で利用されており，教育支援ツールとしての完成度と評価は高い．ハードウェアとソフトウェアを構築することを通して，コンピュータサイエンスにおける理論や応用技術を学ぶことができるものである．まず NAND 回路を用いて論理回路，加算器，ALU と CPU の設計によってハードウェアを構築し，さらにアセンブラとコンパイラ，オペレーティングシステムの設計によりソフトウェアを構築する．これらを 1 から作り上げる作業により，計算機科学に現れる諸概念を理解することができるのである.

[1]は情報工学やその他の工学部系の学部生と大学院生を対象としている．筆者の研究室でも配属された学部生を対象として講義内でも利用している．しかしながら，半期の講義内の限られた時間の中で全部のテーマを学び終えることは難しいのが現状である.これは，学部生の多くが各テーマの初学者であるがゆえに，特に理解するための難易度が高いテーマに多く時間を費やしてしまうことが理由としてあげられる．現状のシステムに加えて効果的な教育支援システムを活用することで，初学者でも難易度の高いテーマを効率よく学習できる環境を提供する必要がある.ここで構築にあたり実装が難しく躓くポイントを以下に提起する.

(1)回路の設計時におけるその実行のタイミングと回路全体の動作の把握が難しい.[3]

(2)アセンブラにおけるワンパスの変換とマルチパスの変換の違い，さらには機械語変換の処理の構造がわかりにくい.[10][11][12]

(3)VM のアーキテクチャの動作とアセンブリ言語へと変換されるその過程がわかりにくい.[4]

(4)コンパイラのフロントエンドにおいて，Jack ソースコードが中間コードである VM コードへと変換されるその過程がわかりにくい.[6][7][8][9]

本研究では，(2)のポイントを取り上げる．これは，アセンブラの設計がコンピュータシステムの中でも難しい部分であり，初めてアセンブラを設計するには補助が必要であるためである．特に，シンボルによる参照をメモリアドレスに置き換えることは簡単ではない．その理由として，ユーザーの定義した変数名やラベルを実際のメモリアドレスへと対応づけなければならないからだ．決められた記号をそれに対応する機械語へと変換する処理はさほど難しくない．しかし，その変換処理をイメージできたとしても，Hack アセンブリ言語の構造や動作の理解が乏しいと作り上げるのは簡単ではない．そのため，本稿では，Hack アセンブリ言語の教育を支援するツールとして，アセンブラの構造や動作を可視化するシステムを提案する．

提案するシステムは，字句解析を用いた生成法を可視化するシステム，正規表現を用いた生成法を可視化するシステムを提供する．字句解析を用いた生成法を可視化するシステムでは，字句解析の部分と構文解析，コード生成の部分の二つの部分に分けて２つのシステムを提供する．字句解析を可視化するシステムでは，字句解析の粗細を用いた解析，動作の可視化を提供する．構文解析，コード生成を可視化するシステムでは，パスの回数の違いによる構文解析，コード生成の処理過程の可視化を提供する．正規表現を用いた生成法では，正規表現，コード生成の処理過程の可視化を提供する．そして，この構造や動作の理解により，前章で実装する Hack アーキテクチャ，Hack 機械語への理解を深めることにも繋がるのである．さらに言えば，CASL Ⅱや PDP-11 のようなよく使われるアセンブリ言語との対応関係を知ることができれば，理解は深められるはずである．

２． Hack コンピュータシステム

計算機科学の諸概念における全体構造を教育支援するシステム(以降 Hack コンピュータシステムと呼ぶ)について述べる．

図2-1. Hackコンピュータシステムの階層

本研究ソフトウェア部(青)とハードウェア部(赤)に分けられる．ハードウェアの構築は仮想的に設計ができるハードウェア記述言語(HDL)で定義する．OS はオブジェクトベースの高水準言語Jack 言語を用いて実現する．また Hack コンピュータシステムに付属する各モジュール(①-⑤)を紹介する．

①HardwareSimlator ―HDL で実装した回路のシミュレーションとテストを行う．

②CPUEmulator ―Hack コンピュータシステムのエミュレートを行う．

③VMEmulator ―バーチャルマシン(VM)の動作をエミュレートする．

④Assembler ―Hack アセンブリ言語をHack 機械語に変換(アセンブル)する．

⑤JackCompiler ―Jack プログラムをVM プログラムに変換(コンパイル)する．

各階層の機能を実現するためにはこれらの定義済みモジュールを利用する．本研究はグループで行っている．自分は上記のツールの中でアセンブラ(④)の部分を担当した．

３． Hack アセンブラ

（１）Hack コンピュータの仕組み

Hack アセンブラを作るには Hack コンピュータの仕組みを理解する必要があるのでその説明を行う．このコンピュータはノイマン型アーキテクチャと概念上の仕組みは同じである．ノイマン型アーキテクチャは中央演算処理装置（CPU）を中心としてメモリデバイスを操作し入力デバイスからデータを受けとり出力デバイスへと送信する．この流れを図 2-2 に示す．

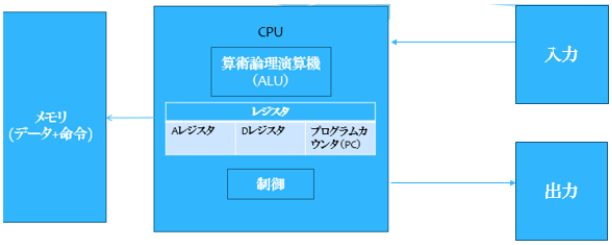

図 2-2. ノイマン型アーキテクチャ

メモリには二種類の情報が格納される．１つはデータ項目であり，もう一つはプログラミング命令であり，それらの情報はそれぞれデータメモリ(RAM)，命令メモリ(ROM)に格納される．CPU は３つのレジスタから構成される．データレジスタとアドレスレジスタとプログラムカウンタである．D レジスタと A レジスタは 16 ビットの汎用的なレジスタであり，算術演算や論理演算において用いられる．D レジスタはデータの保持するために使うが，A レジスタの内容は命令に応じてデータ値，RAM アドレス，ROM アドレスのどれかの１つの意味として解釈される．

（２）Hack アセンブラの仕様

Hack アセンブリ言語の仕様を簡単に説明する．この言語は普通のアセンブラ言語と違い，各コマンドはバイナリ命令と一対一で直接対応する．つまり各コマンドは個別に変換される．よってこの言語のテキストの各行は命令またはラベルのどちらかを示す．コマンドの種類はアドレス命令と計算命令の二種類がある．そしてラベルは格納されるメモリの位置とシンボルを結びつける役割を

する.フォーマットのバイナリは Hack 機械語の仕様に従い 16 ビット命令をベースとしている. A 命令のバイナリは「0vvv vvvv vvvv vvvv」というフォーマットに従い，この命令を実行すればコンピュータは 15 ビットの vv,,,v という値を A レジスタに設定する．C 命令の命令は「111a c1c2c3c4 c5c6d1d2 d3j1j2j3」というフォーマットに従う．a ビットと c ビットは ALU に行うべき関数を指示し，d ビットは ALU の出力を格納する場所を指定する．j ビットは移動条件を指定する．これらはすべて Hack の機械語仕様に従って行われる.

A 命令は@value というフォーマット，C 命令は dest=comp;jump というフォーマットに従っている．A 命令の value の部分は非負数の 10 進数表記またはそのような数字を参照するシンボルを示している．このシンボルは定義済みシンボルか変数シンボルを示す．定義済みシンボルの表を表 3 に示す.

表３　定義済みシンボル

ラベル	RAM アドレス	（16 進数表記）
SP	0	0x0000
LCL	1	0x0001
ARG	2	0x0002
THIS	3	0x0003
THAT	4	0x0004
R0-R15	0-15	0x0000-f
SCREEN	16384	0x4000
KBD	24576	0x6000

変数シンボルはメモリのアドレス 16 から始まりそこから順に割り当てられる．C 命令は３つの領域に分かれていてそれぞれの領域のバイナリ形式へと変換される．ラベルは(xxx)という形で定義され次のコマンドの位置を参照する.

既存のアセンブラを図 3-1 に示す．このツールはソースコードとその出力しか表示されていない．つまりこのアセンブラがどのように動作してバイナリコードを出力したのが明確にはわからないという点が問題点として挙げられた.

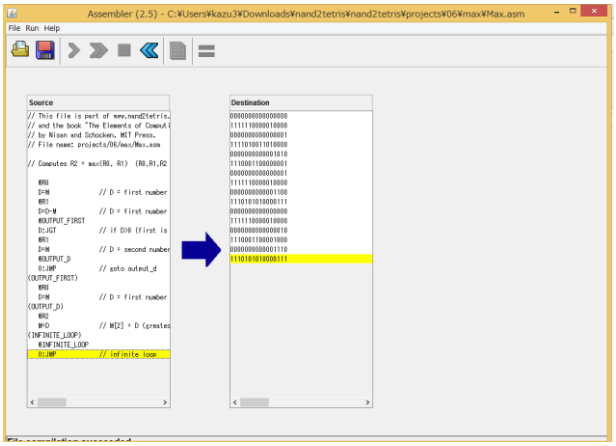

図 3-1. 既存のアセンブラツール

（３）Hack アセンブラの生成法と問題点

a)字句解析を用いた生成法

Hack システムでもこの作り方が推奨されている.この作り方には以下のような処理を行う.

1.アセンブリコマンドを字句解析しトークンに分け，構文解析し、基本となる領域へと分割する.

2.各領域において、対応する機械語のビットを生成する.

3.シンボルによる参照を数字によるメモリアドレスに置き換える.

4.領域ごとのバイナリコードを組み合わせて完全な機械語命令を作成する.

b)正規表現を用いた生成法

1.アセンブラコマンドを一行読み取った後、そこから一文字ずつ読み取り、命令のパターンによって基本となる領域へと分割する．上記の 2 から 4 の処理は同じである.

c)問題点

①字句解析を用いた生成法の問題点

字句解析を行う際に，アセンブラの構造を理解していないと，どのようにトークンに切り分けていいかわからないという点．そして，構文解析の際にも，同じようにどのように解析すればいいかわからないという点が挙げられた.

②正規表現を用いた生成法

アセンブラの構造を理解していないと，命令のパターンを把握することができないという点が挙げられた.

③共通の問題点

二つの作り方の共通の問題点として，シンボルを参照する際に，アセンブラの動作を理解していないとどのように処理していいかわからないという点が挙げられた.

４．システムの設計

システムの設計をする上で，必要になる重要な要素を以下に示す.

（1）二種類の字句解析

字句解析を行う際に，二種類の字句解析の方法を提示することにより，Hack アセンブラの構造の理解が深まると考えた．具体的には，シンボルを省略しない字句解析とシンボルを省略し予約語を準備する字句解析に分けた．以後，前者を細かい字句解析，後者を粗い字句解析と呼ぶ．

これらのトークンの分け方を表に示す．分け方のルールとして，扱った拡張 BNF は付録 1 を参照．

表 4.1　二種類のアセンブラのトークンの分け方

細かい字句解析の分け方	粗い字句解析の分け方
①. 記号	①. 記号
②. ラベルシンボル	②. ラベルシンボル or 定義済みシンボル or ニーモニック
③. 変数シンボル	③. 変数シンボル
④. 定義済みシンボル	④. 定数
⑤. 定数	
⑥. ニーモニック	

粗い字句解析での実際のプログラムでの分け方は，記号，大文字列，小文字列，整数に分ける．その後，大文字列とあらかじめ用意しておいた予約語を比較するやり方になっている．

（2）有限オートマトンの状態遷移図

Hack アセンブラの字句解析の動作を有限オートマトンの状態遷移図で表記した．その全体図をそれぞれ図４-1，2 に示す．

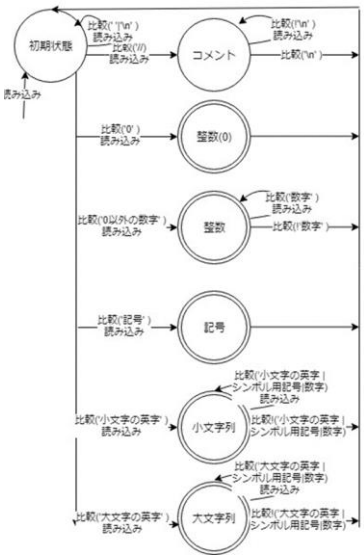

図 4-1．粗い字句解析の有限オートマトンの全体図

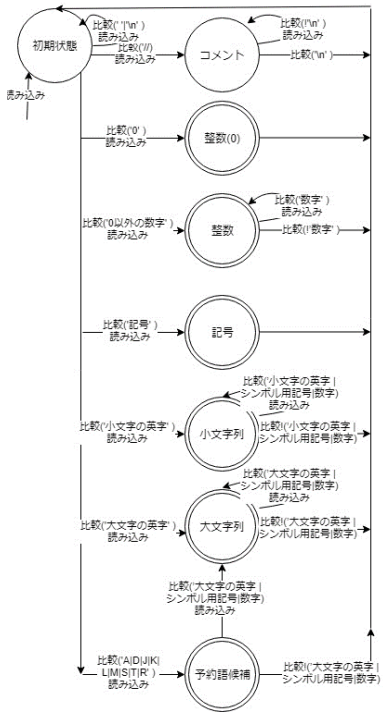

図 4-2. 細かい字句解析の有限オートマトンの全体図

→の途中にどのような動作を行っているかを表記している．受理状態から初期状態に遷移した時点で，トークンの切り分けを行う仕様になっている．

（3）　正規表現の図化

Hack アセンブラの構造は上記の通り，一般的アセンブリ言語より，簡単な構造になっている．そのため，命令のパターンを正規表現で表記することが可能である．その詳細は付録 3 に参照．

５．システムの詳細

本システムは，上記で示した 3-3.3 の問題点に対しての解決策として，アセンブラの構造や動作を可視化するシステムを提案する．

（1）字句解析を用いた生成法を可視化するシステム

a)字句解析の構造や動作を可視化するシステム

a-a)入力と出力

字句解析の入力は Hack アセンブラソースコードとする．出力はトークンとして分けられたファイルとする．

a-b)可視化の構成

入力されたソースコード，解析時に変化する表，有限オートマトンの状態遷移図，字句解析の種類変更ボタンが可視化される．

a-c)解析単位

・一文字ごとに実行

入力されたソースコードを一文字ごとに実行する．

・最後まで実行

ソースコードがトークンに分けられるまで解析する．

a-d)解析方法と可視化

入力されたソースコードを一文字ずつ解析する．読み取った一文字を拡張 BNF（付録 1）に沿って，トークンに分けていく．トークンの分け方は，整数トークン，記号トークン，小文字列トークン，大文字列トークン，予約語トークンに分けられる．切り分け方として，４－１で示したように，粗い字句解析と細かい字句解析の両方の字句解析をできるように用意した．

切り分ける際に，有限オートマトンの現在の状態，現在読み取っている文字，現在予測されるトークンの文字列を表に格納していく．そして，それに従って有限オートマトンの状態遷移図も表示する．

画面遷移が起きる場合，一画面では表示できないため，二画面を表示することにした．

b)構文解析，コード生成の構造や動作を可視化するシステム

上記の字句解析の分けた方に対応した構文解析，コード生成を考える．シンボルの参照を主として，シンボルテーブル,行数カウントを可視化させた.来られにより，シンボルの参照の動作を可視化した．そのシステムを提案する．以下に可視化画面と実行の詳細を示す．

b-a)入力と出力

入力は字句解析で出力されたトークンファイルとする．出力は入力された Hack アセンブラコードを機械語へ変換したコードとする．

b-b)可視化の構成

Hack アセンブラソースコード, 入力されたトークンファイル，予測図，変換される機械語，行数カウント，シンボテーブル, パスの回数変更ボタンが可視化される．

b-c)解析単位

・一トークンごとに実行

入力されたトークンコードをワントークンごとに実行する．

・最後まで実行

ソースコードが機械語に変換されるまで解析する．

b-d)解析方法と可視化

入力されたトークンコードをワントークンずつ解析する. 読み取ったトークンを拡張 BNF（付録３）に沿って，構文解析を行っていく．その際に，改行ごとに行数カウントを増やす．そして，ワントークンごとに，次に予測されるトークンの予測図を可視化する．詳細は付録５を参照．また，解析方法として，パスの回数を分けられる構文解析を用意した．

（２）正規表現を用いた生成法を可視化するシステム

Hack アセンブラの構造は上記の通り，一般的アセンブリ言語より，簡単な構造になっている．しかし, [1]には，字句解析の分けた方については詳しく記されていない．そのため，字句解析を行う際に，どのように分けられているかを可視化するシステムを提案する．以下に可視化画面と実行の詳細を示す．

a)入力と出力

入力は Hack アセンブラコードとする．出力は入力された Hack アセンブラコードを機械語へ変換したコードとする．

b)可視化の構成

Hack アセンブラソースコード，正規表現の図，変換される機械語，行数カウント,シンボルテーブルが可視化される．

c)解析単位

・一行ごとに実行

入力されたソースコードを一行ごとに実行する．

・最後まで実行

ソースコードが機械語に変換されるまで解析する．

d)解析方法と可視化

入力されたソースコードを一行ずつ解析する．読み取った行を正規表現（付録２）に沿って，解析していく．その際に，改行ごとに行数カウントを増やす．そして，一文字ごとに，次に予測されるトークンの正規表現の図を可視化する．一文字ずつ解析するごとにその正規表現の図の画面を遷移させる．一文字ずつ解析するごとにその正規表現の図の画面を遷移させる．

６．一般的なアセンブリ言語との関係

一般的なアセンブリ言語とは，CASLⅡや PDP-11 のようなメジャーなアセンブリ言語である．これらの対応を示すため，CASLⅡから Hack アセンブリ言語への変換を行った. COMMETⅡには三種類のアドレッシングモードしかない. ADDA を例として考える．

例)．ADDA g,x,X

x＝0 の時，絶対番地モード，x≠0 の時，レジスタモード，X がある時，インデックスモードとなる．それぞれのモードを Hack アセンブリ言語に変化した表を下記に示す．

表 6-1. 各モードと Hack アセンブリ言語の対応

レジスタモード	インデックスモード	絶対番地モード
@x D=M @g M=D	@x D=M @X D=D+M @g M=D+M	@x D=M @g M=D+M

CASLⅡの各命令を三種類に分けた．分け方としては，Hack アセンブリ言語に変換した際に，同じ命令の構造になるものにしている．

表 6-2 各命令の分け方

1	LD 命令，算術・論理演算命令，比較演算命令
2	LAD 命令，シフト命令，分岐命令,ST 命令
3	スタック命令，コール命令，リターン命令，その他

表 6-1,2 により，CASLⅡの全ての命令から Hack アセンブリ言語への変換をすることができる．

7．システムの実現

開発環境は Java と JavaFX を使用する．

（1）字句解析を可視化するシステムの実装

本研究で実装する字句解析器は，一文字ずつ字句解析を行い，状態遷移図を用いて字句解析の動作を可視化する．以下にプログラムの構成を分けたものを示す．

1．アセンブリ言語の字句解析器が受理する字句要素（付録 1）をリストに格納する．

2．読み込んだコードを一文字ずつに分け，一文字ずつリストに格納する．

3．読み込んだコードを一文字ずつ(1)のリストと比較し解析結果を返す．

4．その解析の動作を状態遷移図で表示する．

字句解析のやり方として，二種類の字句解析を用意した．可視化画面を図 7-1, 2 に示す．プログラムの詳細は付録 4 を参照．

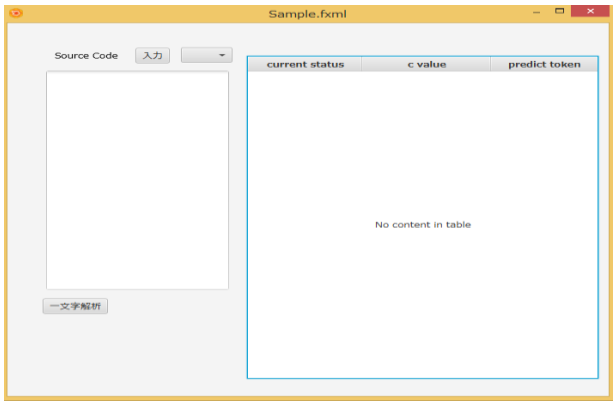

図 7-1.字句解析の可視化システムの画面 1

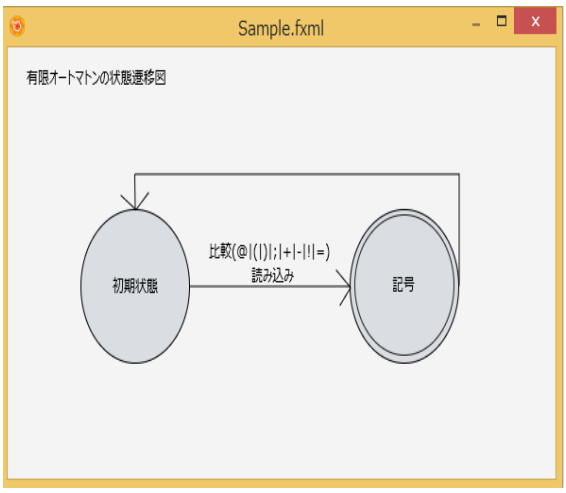

図 7-2.字句解析の可視化システムの画面 2

（2）構文解析，コード生成を可視化するシステムの実装

本研究で実装する構文解析，コード生成を可視化するシステムは，パスの回数を分けて実装されている．

a) ワンパスの実装方法

プログラム全体をみていき，構文解析，コード生成をしながらシンボルの参照の動作を可視化する．以下にプログラムの構成を分けたものを示す．

1．アセンブリ言語の構文解析器が受理するトークン要素（付録 1）をリストに格納する．

2．読み込んだコードはトークンに分けられているため，一トークンずつリストに格納する．

3．読み込んだトークンを一トークンずつ(1)のリストと比較し解析結果を返す．

4．その結果を基に機械語へと変換していく．そして，改行が起こった場合，行数カウントを 1 増やす．この動作を最後まで実行する．その際に，シンボルがあった場合，シンボルテーブルにそのシンボルと対応するアドレスを格納し，機械語へと変換する．

4‘．4 の際に，シンボルがあった場合，シンボルテーブルにそのシンボルと対応するアドレスを格納し，機械語へと変換する．

4‘’．4‘の際に，シンボルがユーザー定義ラベルかつそのラベルの該当する疑似コマンドがそのラベルより前に定義されている場合，不確定ラベルとして，そのラベルの行数とラベル名を別の場所に格納しておく．そして，そのラベルの疑似コマンドが実行された時に，該当のラベルに対応するアドレスをシンボルテーブルに格納する．

この可視化画面を図 7-3 に示す．プログラムの詳細は付録 5 を参照．

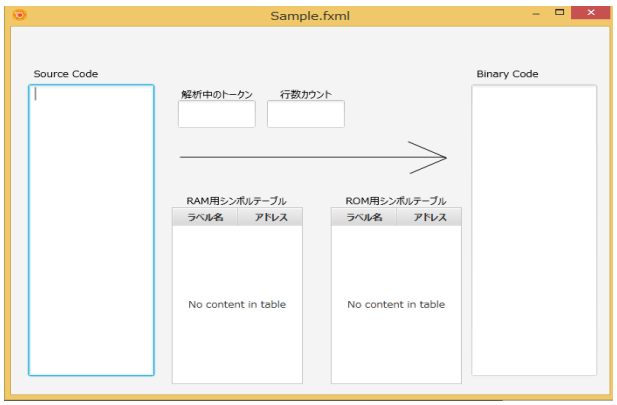

図 7-3.ワンパスの可視化画面

b)ツーパスの実装方法

１回目のパス：プログラム全体をみていき，RAM のシンボルテーブルを完成させる．行数をカウントする．

２回目のパス：コード生成をしながら，ROM のシンボルテーブルを完成させ，シンボルを対応するアドレスに変換する．

この可視化画面を図 7-4 に示す．

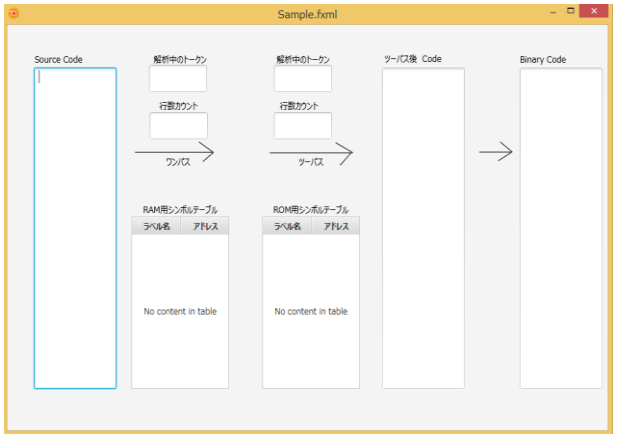

図 7-4.ツーパスの可視化画面

（３）正規表現を用いた生成法を可視化するシステムの実装

１回目のパス：プログラムを一行ごとに解析していき，RAM のシンボルテーブルを完成させる．行数をカウントする．

２回目のパス：正規表現によるパターンマッチングを用いながら，図を表示しながら解析していく．コード生成をしながら，ROM のシンボルテーブルを完成させ，シンボルを対応するアドレスに変換する．

この可視化画面を図 7-5 に示す．正規表現の図を図 7-6 に示す．プログラムの詳細は付録 6 を参照．

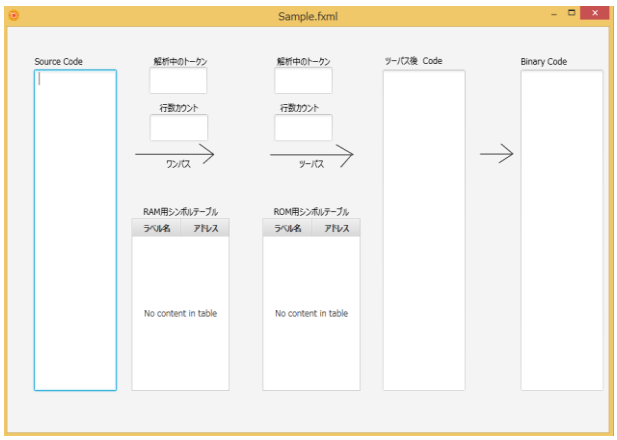

図 7-5.正規表現を用いた生成法の画面

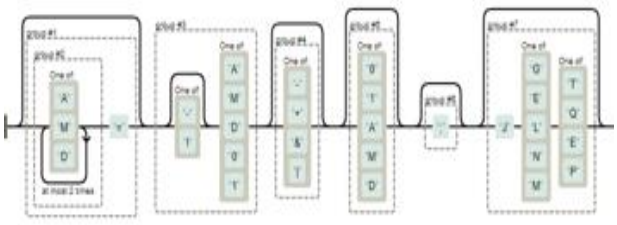

図 7-6.正規表現の図

８．今後の課題

本稿で実現した可視化システムを本研究室の講義である PBL で使用し，現状のツールとの比較・評価を行う．今回扱った Hack アセンブリ言語ではない一般的なアセンブリ言語のようなどんな言語にも対応するようなシステムの実現が課題である

９．まとめ

本稿では[1]のアセンブラの学習に関して，初学者でもわかりやすく学習できるように，基礎の学習としてアセンブラの構造や動作の理解を図った．その理解の補助として，アセンブラを可視化するシステムを提案している．自分自身としてはこの研究を通じて本の理解，つまり本の目的であるコンピュータサイエンスの重要なテーマを学ぶことができた．この本を読むことによりコンピュータサイエンスの全体の流れを把握することができる代物になっている．実際に手でプログラミングをすることにより明確にその章のテーマを理解することにつながる．この実装がさらにわかりやすく使いやすくなればよりこの本を理解することができるだろう．そのため，この研究を通じてそれが少しでも実現できていたら幸いである．この経験を活かして研究とはまた別のなにかにつながるようなことができれば自分としても幸いである．

謝辞：本研究に関して，様々なご指導を頂きました和田幸一教授に深く感謝いたします．また，様々な質問，疑問点に真剣に対応していただいた，和田研究室の同期，大学院生の皆様に深く感謝しております．この研究が少しでも誰かの役に立ったら幸いです．

付録 1~6

参考文献

1) N. Nisan and S. Schocken(著), The elements of Computing Systems: Building a ModernComputer from First Principles, The MIT Press, 2005.

2) Noam Nisan, Shimon schocken(著), 斉藤康毅(訳)
「コンピュータシステムの理論と実装　モダンなコンピュータの作り方」
オーム社

3)猪狩淳,"Hack 教育支援システムにおける順序回路の可視化について", 法政大学理工学部応用情報工学科卒業論文, 2019.

4)大村優斗,"Hack 教育支援システムおける仮想計算機の可視化ツールについて", 法政大学理工学部応用情報工学科卒業論文, 2019.

5)久保田祐貴, 和田幸一,"Hack システムにおけるコンパイラの教育を目的とした教育支援システムの改良とその実現" 2018 年度電子情報通信学会総合大会学生ポスターセッション, 2018.

6)久保田祐貴, 和田幸一,"Hack システムにおけるコンパイラの教育を目的とした教育支援システムの改良とその実現"2019 年度電子情報通信学会総合大会, 2019.

7) 久保田祐貴, "計算機科学における諸概念に対する教育支援システムに関する研究",法政大学大学院理工学研究科応用情報工学専攻修士論文,2019

8]中村光希, ”教育支援を目的とした Hack システムの LL(1)構文解析における Director 集合を求める過程の可視化の実装”, 法政大学理工学部応用情報工学科卒業論文, 2020.

9)小寺遼, ”教育支援を目的とした JACK 言語における字句解析可視化ツールの実装”,　法政大学理工学部応用情報工学科卒業論文, 2020.

10)岩本和也, 和田幸一, "Hack システムにおけるアセンブラの教育とその作成方法を支援するツールについて"2019 年度電子情報通信学会総合大会学生ポスターセッション,2019.

11) 岩本和也, 和田幸一, "Hack システムにおけるアセンブを支援するツールの実装"2020 年度電子情報通信学会総合大会学生ポスターセッション,2020.

12) 岩本和也, "Hack システムにおけるアセンブラの教育とその作成を支援するツールについて", 法政大学理工学部応用情報工学科卒業論文,2018.